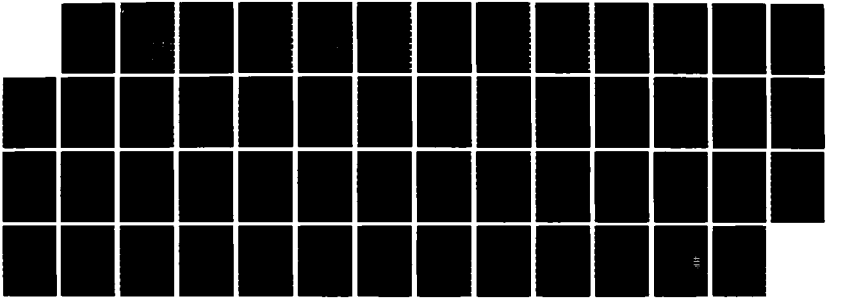
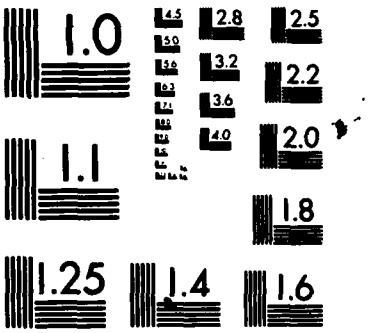
AD-A171 267 GENIE: AN INFERENCE ENGINE WITH APPLICATIONS TO VULNERABILITY ANALYSIS(U) ARMY BALLISTIC RESEARCH LAB ABERDEEN PROVING GROUND MD F S BRUNDICK ET AL. JUN 86 BRL-TR-2739 1/1

UNCLASSIFIED F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

TECHNICAL REPORT BRL-TR-2739

GENIE: AN INFERENCE ENGINE WITH APPLICATIONS TO VULNERABILITY ANALYSIS

Frederick S. Brundick
John C. Dumer
Timothy P. Hanratty
Paul J. Tanenbaum

June 1986

US ARMY BALLISTIC RESEARCH LABORATORY
ABERDEEN PROVING GROUND, MARYLAND

Destroy this report when it is no longer needed.
Do not return it to the originator.

Additional copies of this report may be obtained from the National Technical Information Service, U. S. Department of Commerce, Springfield, Virginia 22161.

The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

The use of trade names or manufacturers' names in this report does not constitute indorsement of any commercial product.

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER Technical Report BRL-TR-2739	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) GENIE: An Inference Engine with Applications to Vulnerability Analysis		5. TYPE OF REPORT & PERIOD COVERED Final 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Frederick S. Brundick Paul J. Tanenbaum John C. Dumer Timothy P. Hanratty		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS U.S. Army Ballistic Research Laboratory ATTN: SLCBR-VL Aberdeen Proving Ground, MD 21005-5066		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Ballistic Research Laboratory ATTN: SLCBR-DD-T Aberdeen Proving Ground, MD 21005-5066		12. REPORT DATE June 1986 13. NUMBER OF PAGES 52
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Expert Systems
Artificial Intelligence
Vulnerability Analysis
Knowledge Representation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

An expert system is a program that mimics the performance of a human expert in some decision making process. We are building expert systems to deal with vulnerability analysis and have developed a general-purpose mechanism (called an "inference engine") with which to emulate the behavior of human experts. The project is called "Genie", and this report offers a presentation of its design, its implementation, and its usage. Genie is primarily a goal-driven (backward chaining) engine, but whenever it makes a deduction it is used wherever possible in data-driven mode (forward chaining). Production rules

are written in simple everyday English; a preprocessor in Genie converts these rules into a frame-based knowledge representation scheme for efficient computation. We

We will then treat several facets of the interface between Genie and its human users, including Genie's facilities for explaining and justifying its behavior. Finally, we ~~will look~~ at the first application of Genie by the Army in a Turbine Engine Expert System and present some of the additions and improvements we plan to implement.

An inference engine shell will aid in the development of future expert systems, while the expert systems themselves will make the vulnerability analysts' jobs much easier.

TABLE OF CONTENTS

TABLE OF CONTENTS (cont'd)

LIST OF ILLUSTRATIONS

I. INTRODUCTION

A. Background

Determining the vulnerability of combat vehicles and other systems to the wide range of threats presented in modern warfare can be a daunting task. Although physics, engineering, operations research, and other disciplines offer an invaluable framework for addressing the problem, these more-or-less hard sciences cannot by themselves provide complete solutions. On the contrary, practitioners of vulnerability analysis will assure you that their profession is at least as much an admixture of art, intuition, and educated guesswork as a science.

In this regard, however, vulnerability analysis is not markedly different from contemporary medicine, analytical chemistry, or petroleum exploration. These are just a few of the occupations that are believed by the layman to be (and, too, are often presented in textbooks as being) soundly based in scientific theory and empirical background, though they actually rely to a large degree on non-rigorous hunches, personal experience gleaned over long careers, and the occasional dose of black magic. In all these fields, with the emphatic inclusion of vulnerability analysis, the computer has been of tremendous assistance in tackling those segments of the job that can be successfully addressed with mathematical and scientific formalisms.

Another similarity among these fields is the unfortunate inability of traditional computer techniques to master the entire problem. Many a human expert has been heard to lament, "Crunching numbers is fine, but computers lack the smarts, the rationality to handle this problem." It was to address this harder class of problems that the artificial intelligence (AI) community developed the technology that has come to be called *expert systems*. A true expert system is a program that mimics the performance of a human expert in some intellectual endeavor. The archetypal expert system attains this high level of proficiency by embodying the heuristic, informally framed knowledge of the human expert, along with the expert's not-always rigorous methods of reasoning in the subject domain.

We are building expert systems to deal with vulnerability analysis. Toward this end, we have developed a general-purpose mechanism (called an *inference engine* in AI terminology) with which to emulate the behavior of human experts. The project, developed in Franz LISP on a VAX 11/750, is called *Genie*, and this report offers a presentation of its design, its implementation, and its usage.

As our first application of Genie, we are working with Walter Thompson and Steven Polyak of the Aerial Targets Branch, Vulnerability/Lethality Division, on an expert system to assist human experts in assessing the vulnerability of turbine jet engines. Many of the examples cited in this report can be thought of as being taken from such a system, although authenticity of domain details will sometimes suffer for the sake of illustrative clarity. For an authentic run of the turbine engine expert system, called *Tess*, see Appendix A.

B. Typical Architecture of an Expert System

The highest level of organization of an expert system is not complex, as Figure 1 illustrates.[1]

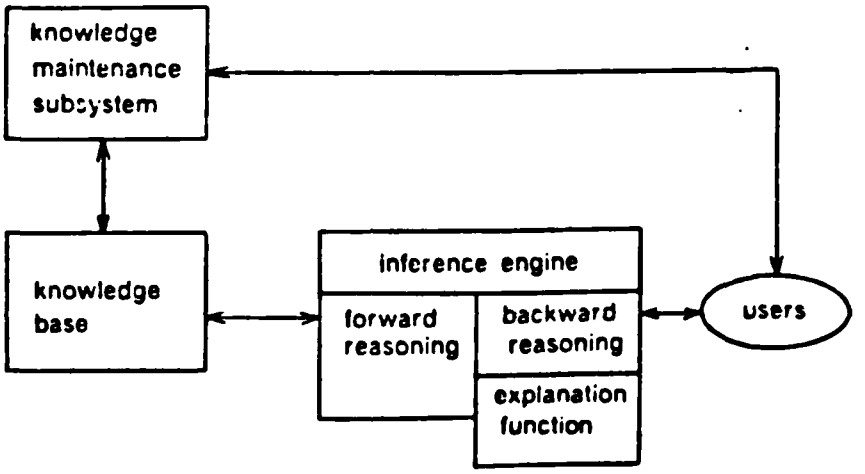


FIGURE 1.—*System Overview*

A basic tenet of the expert system methodology requires separation of domain expertise (whether it be in infectious diseases or turbine engine vulnerability) from the strategies for manipulating and applying that expertise. The resultant modularity greatly facilitates debugging and modifying either subsystem.

The first of the subsystems is called the ***knowledge base***. It is the mass of expertise that has been collected from one or several human experts. The

---

[1] Adapted from K. Niwa, K. Sasaki, and H. Ihara. "An Experimental Comparison of Knowledge Representation Schemes", *AI Magazine*, Summer 1984. pg. 30.

knowledge base will typically contain both declarative knowledge (e.g. that the specific gravity of steel is 7.8) and procedural knowledge (e.g. that a means of determining the specific gravity of an arbitrary solid is to calculate the ratio of the solid's density to the density of water). Procedural knowledge is often expressed in rules of propositional implication (e.g. that if a solid is known to be roughly as dense as steel then it can be deduced that the solid's specific gravity is approximately equal to that of steel). The process of constructing a knowledge base, of determining how an expert solves problems and implementing that knowledge on a computer, is called ***knowledge engineering***.

The second subsystem is called the ***inference engine***. In a sense, the inference engine is the program that acts upon the knowledge base as data. Optimally, it would be so well separated from the knowledge base and so general that one could plug in a different knowledge base and thereby create a new expert system in an unrelated domain.

Though it might not be as fundamental, an interface between the system and its users is no less important than the knowledge base or the inference engine. The user interface should allow a non-programmer to modify the knowledge base, obtain consultative assistance, and interrogate the system about the reasons for its behavior, all in language that is clear and natural to the user.

II. ORIGINS

Genie is greatly evolved from its earliest sources. But their influence can still be distinguished, at least on the level of concepts and principles. The earliest form of the system's control mechanism was based on Animal, a prototype expert system developed by Patrick Winston and Berthold Horn[2].

Matthew Rosenblatt of the Army Materiel Systems Analysis Activity added several simple natural-language capabilities to Animal. As an example, Winston's routine requests user verification of a fact† by simply displaying the fact (e.g. "Is this true: object has low specific gravity"), whereas Rosenblatt's version could turn the fact into a question (e.g. "Does object have low specific gravity?"). Genie has incorporated these additional capabilities into its user interface.

---

2 P. H. Winston and B. K. P. Horn. *LISP.* Addison-Wesley, 1981. pp. 240-249.

† Here *fact* is not used in the colloquial sense of an assertion that is known with certainty to be true. Instead, a fact is a proposition that may be true or false, or may have no truth value at all in particular circumstances.

III. REPRESENTATION OF KNOWLEDGE

A key subproblem in modeling the behavior of a human expert is to choose appropriate formats for expressing and storing the expert's knowledge so that a computer can process the knowledge effectively and efficiently. In this section we discuss the approach to knowledge representation that has been employed in building Genie. The discussion is presented in two parts: first we consider the syntax of our representation scheme, the formats available for use; and then we consider their semantic content, what mental constructs these structures represent.

A. Tools

1. Frames. The data structure that is most widely used by Genie is called a *frame*. First popularized by Marvin Minsky[3], frames are a method of organizing information about an object and its properties, together with pointers to other frames describing related objects. A frame is composed of *slots*, each of which contains a piece of information. Figure 2 illustrates the kinds of information that might be stored in a frame.

The sample frame in Figure 2 contains information about an object known to us as *Comp.T58*, the compressor section of the T58 turbine engine, so we refer to the frame as **Comp.T58**†. The **A Kind Of** slot — often abbreviated **AKO** — specifies the class of objects to which Comp.T58 belongs. A knowledge base might include a separate frame, called **compressor**, that would contain knowledge about compressors in general. This would improve storage efficiency, since general knowledge would not need to be stored in **Comp.T58** and then duplicated every time we built a frame for the compressor of a new engine. More importantly, though, this would better capture the intrinsic order in the knowledge. If a required piece of information were not found in **Comp.T58**, one could proceed to the **compressor** frame, and, if necessary, even higher up this AKO hierarchy.

---

3 M. Minsky. "A Framework for Representing Knowledge", in *The Psychology of Computer Vision*. P. H. Winston, ed. 1975. pp. 211-275.

† In this discussion names of frames, slots, and facets will appear in **Boldface**.

Comp.T58	
A Kind Of	compressor
Type	axial
Manufacturer	General Electric
Found In	T58 turbine engine
Pressure Ratio	8.3 : 1
Air Mass Flow	5.6 kg/s
Blade Material	steel
Exit Pressure	calculate: intake pressure times pressure ratio

FIGURE 2.— *Typical Frame*

Another benefit of arranging frames in an AKO hierarchy can be seen in the variety of slots in our example. When one builds a frame for a new instance of some object class, what should the form of that frame be? It is plain that a frame describing a shaped charge munition should contain few of the slots found in **Comp.T58**. One solution to the problem is to go to the frame representing the appropriate class of objects and fetch a template. The **compressor** frame, for example, could specify the slots that should appear in each instance frame.

The **Exit Pressure** slot in **Comp.T58** contains a piece of procedural knowledge. This knowledge takes the form of a function called an *if-needed daemon*, an operation which is to be performed if a value of exit pressure is ever required. Daemons can be used for many other purposes, too, such as performing an operation whenever a frame is modified. It should be noted that, for the sake of hierarchical knowledge organization, the **Exit Pressure** slot might be better located in the general **compressor** frame. It is included here for illustrative purposes only. Similarly, the **Manufacturer** slot probably belongs in the frame for the T58 engine. There are many other ways that this information might be organized into frames — the best choice depends on the application. The flexibility of the frame approach is apparent.

The **Comp.T58** frame is similar in meaning to the structures used in Genie that are called *concept frames*. As the name suggests, a concept frame is

intended to embody the system's concept of some object, everything it knows about the object. Concept frames will be discussed in the section on semantics below.

Of course, if frames are to be used, then special functions will be required to manipulate them. We have implemented a set of routines to retrieve and store information, and to perform other such tasks in the manner described by Winston and Horn[4] and by Roberts and Goldstein[5].

2. Value Ranges. Arithmetic manipulation of data is a central theme of computer science. Expert systems do not manipulate numbers as often as they do more general symbols, but a means to perform numeric computation is unquestionably among Genie's requirements. Unfortunately, the kinds of numeric knowledge which Genie must handle are characterized by the same imprecision and uncertainty that first motivated the development of expert systems. There are in the literature many approaches to dealing with the problem of imprecise and uncertain knowledge, some of which will be discussed in section VIII below. Another representation problem can turn up even with knowledge that is perfectly precise, crisp, and certain. Some knowledge is inescapably cast as information about sets, vectors, or intervals, rather than scalars.

We have developed a representation format to deal with both of these problems. Using what we call *value ranges* and the routines for manipulating them, one can represent either intervals on the real line or imprecisely known scalars. A value range record consists of six cells which can be labeled as in Figure 3.

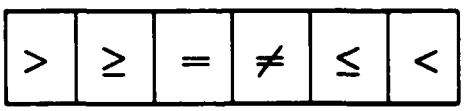


FIGURE 3.—*Format of a Value Range Record*

Either of the first two cells can be used for a lower bound on the value. Similarly, the last two cells can bound the value above. The $\neq$ cell can contain a set of prohibited values. The $=$ cell can only hold one value; if it is non-

---

4 Winston and Horn. op. cit. pp. 291-301.

5 R. B. Roberts and I. P. Goldstein. *The FRL Primer.* Memo No. 409, Artificial Intelligence Laboratory, MIT, 1977.

empty, then all the other cells must be empty. As an example, the value range in the proposition that

$$0 < x \leq 10, \quad x \text{ not} \in \left\{\frac{\pi}{2}, 6, 7\right\} \tag{1}$$

would be represented by the record

0			$\left\{\frac{\pi}{2}, 6, 7\right\}$	10	

FIGURE 4.—*Inequality* (1) *Represented as a Value Range*

One way that Genie uses value ranges is in representing rules in the knowledge base. For example, a knowledge base might contain the following rule,

"If thrust of engine < 6000 lbs, or
thrust of engine > 12,000 lbs,
then engine is not J57."

Another use for value ranges is Genie's facility for interpreting a user's answers to its questions. In answer to the question, "What is the thrust of engine?" the user may enter, "5000 <= thrust < 7500". There are several functions that operate on value ranges, including routines to build a record, to determine if a hypothesized value conflicts with existing knowledge, and to improve the precision of the system's estimate of a value.

B. Semantics

For the most part, Genie manipulates knowledge that is expressed in propositional implications known as *production rules*, or *productions*. A production links one group of propositions, called its *antecedents*, to a second group of propositions, called *conclusions*. The individual propositions are represented by fact frames, and the productions by rule frames. A third type of frame used by Genie is the concept frame. All three frame types have two states. The static state consists of intrinsic information and relations; it is run-time invariant. The dynamic state adds the results of all the manipulations performed during the current execution.

1. Rules. In practice, a rule frame often contains just a list of preconditions for applying the rule together with a list of deductions that result from its application. These are stored in the **Ifs** and **Thens** slots, respectively. Since a proposition and its negation describe the same knowledge, we use one fact frame to represent both. Therefore, propositions appearing in rules must be marked to specify which sense should be used. As an example, consider rule6, whose frame appears in Figure 5a. It states that if fact19 and fact20 are known to be true and fact3 is known to be false then fact21 is true and can be added to the knowledge base.

Once the three preconditions are met and the deduction is made, **rule6** takes the form given in Figure 5b. In this case, rule6 is said to have *fired*. If, however, fact20 were determined to be false, the rule would not fire and nothing would be learned about fact21's truth value. This situation is illustrated in Figure 5c.

Flexibility in specifying a rule's preconditions is provided by the must~have mechanism shown in Figure 5d. The **rule88** frame contains three antecedents. The **Must~have** slot indicates that if any two of the antecedents can be determined to hold, then the rule can fire. The default, for rules without **Must~have** slots, requires all the preconditions to be met.

2. Facts. The fact is the basic semantic building block in a Genie knowledge base. Rules are built up from them, and the inference engine attempts to deduce or verify them. When the user is asked a question, it is in order to acquire new facts. And the system's ultimate answer is some fact. The internal representation of facts is illustrated in Figure 6.

The frame **fact17** (Figure 6a) represents a proposition about a compressor. The English-language statement of that proposition is found in the **Stmt** slot. The **Default** slot is available to specify which sense of a proposition (viz. its affirmation or its negation) should be assumed in the event that its truth value cannot be determined directly. The **Ifs~of** slot lists all the rules in the knowledge base whose application depends upon the truth value of fact17. In particular, fact17 is in the **Ifs** slots of rules five, twenty-seven, and twenty-eight. The **Thens~of** slot, on the other hand, lists all the rules that, if applied, will determine fact17's truth value: fact17 is in the **Thens** slots of rules 16 and 105. This linking of facts to the rules that use them speeds execution and facilitates explaining system behavior to the user.

The **Xor** slot of **fact17** contains the name of a group of mutually exclusive propositions. If, by whatever means, fact17 is determined to be true, then

rule6	
Ifs	(fact19 1) (fact20 1) (fact3 0)
Thens	(fact21 1)

(a)

rule6	
Ifs	(fact19 1) (fact20 1) (fact3 0)
Thens	(fact21 1)
Status	Fired

(b)

rule6	
Ifs	(fact19 1) (fact20 1) (fact3 0)
Thens	(fact21 1)
Status	Failed
Culprit	fact20

(c)

rule88	
Ifs	(fact15 1) (fact32 0) (fact33 1)
Thens	(fact34 1) (fact5 0)
Must~have	2

(d)

FIGURE 5.—*Sample Rule Frames.*
(a) *The static version of a simple rule.*
(b) *The dynamic version of a successful application of* (a).
(c) *A dynamic version of a failed application of* (a).
(d) *A must~have rule.*

fact17	
Stmt	Stator vanes are aluminum
Ifs~of	rule5 rule 27 rule 28
Thens~of	rule16 rule105
Xor	xor3
Default	Stator vanes are not aluminum

(a)

fact17	
Stmt	Stator vanes are aluminum
Ifs~of	rule5 rule 27 rule 28
Thens~of	rule16 rule105
Xor	xor3
Default	Stator vanes are not aluminum
Truth	True
How	Deduced using rule16

(b)

fact23		
Stmt	Speed of spool GT 12000	
Ifs~of	rule18 rule120	
Arithmetic	concept	spool
	attr	speed
	relat	GT 12000
Truth	False	
How	Num-Relat	

(c)

FIGURE 6.—*Sample Fact Frames.*
(a) *The static version of a simple fact.*
(b) *A dynamic version of* (a).
(c) *A dynamic version of an arithmetic fact.*

Genie will conclude that all the other facts in xor3 are false. This feature is a handy way to model hierarchies of disjoint classes of objects (e.g. Whether a target is of rolled homogeneous armor, mild steel, or aluminum). It also provides a means of representing the relation between antonyms, as for example, whether compressor vane geometry is variable or fixed.

Suppose Genie were running and applied rule16. As a result, it would amass dynamic knowledge about the truth value of various facts. If rule16's assertion about fact17 were in the affirmative, then **fact17** would be modified to the form shown in Figure 6b. The **How** slot could be checked at a later time to determine the context in which fact17's truth was determined.

Propositions concerning numeric knowledge can be represented using what we call *arithmetic facts*, as illustrated in Figure 6c. The **Arithmetic** slot of an arithmetic fact frame has three sub-cells (called *facets* in frame terminology), which provide the links by which to confirm or disprove fact23. According to the **concept** and **attr** facets, fact23 concerns the attribute called speed of an object called spool. Genie determined that fact23 was false by looking in the **spool** frame to compare what it knew about spool speed with the relation stored in the **relat** facet of **fact23**.

3. Concepts. Rule and fact frames provide a significant increase in deductive speed, but they are little more than an extension of the production-rule approach to building expert systems. One of the major advantages of production rules over other semantic structures is their modularity. Rules represent small pieces of knowledge and can be added to a knowledge base or modified easily, and undesired side-effects are less common than with more complex structures. Unfortunately, though, the modularity of production rules also represents one of their most serious drawbacks. Because a knowledge base made up of rules has such a fine granularity, it is difficult to ascertain high-level patterns and order in the knowledge. This problem is especially serious when a user tries to understand the system's lines of reasoning, or when a student tries to acquire the expertise inherent in the knowledge base.[6]

As a further step towards representing the order in an expert's knowledge, Genie uses concept frames to group information. In the discussion of

---

[6] A. Barr and E. A. Feigenbaum, eds. *The Handbook of Artificial Intelligence*, vol. 1. William Kaufmann, Inc., 1981. pp. 193-194.

arithmetic facts, above, we saw one circumstance in which concept frames are used. Proceeding with that example, we shall trace Genie's process of determining the truth value of fact23. The system's knowledge about several of the spool's attributes are stored in **spool**, which might take the form shown in Figure 7.

spool							
speed	value~range			9590			
	used~by	fact23 fact50 fact82					
	asked	Yes					
	value	9590					
length	value~range						
	used~by	fact40 fact95					
mass	value~range	0				500	
	used~by	fact43 fact44 fact60 fact99					
	asked	Yes					

FIGURE 7.—*Sample Concept Frame.*

Originally, there was insufficient information in the **speed** slot of **spool** to determine whether fact23 held. So Genie asked the user, "What is the speed of spool?" The user's response was presumably the scalar value 9590, since that is what the **value~range** facet indicates. When the =cell of a **value~range** is non-empty, its contents are also stored in the **value** facet.

Neither fact40 nor fact95 has been needed yet, since the **length** slot retains its static configuration. But one of the four propositions about spool mass was needed, since the **asked** facet of **mass** is full. If one of the three remaining facts asserted that spool mass equaled 200, then Genie would be incapable of determining that fact's truth value by direct calculation. This is because Genie will only request input of a given parameter once, on the assumption that the user will have given his best estimate immediately.

IV. REASONING METHODS

The basic intent in developing expert systems is to enable a computer to reason as a human expert does. By this we mean *to achieve expert performance*, since our goal is not so much modeling the expert's behavior as modeling the results of that behavior. A parallel can be drawn with the expert system's representation of knowledge: one seeks a format that is isomorphic to the one used by the expert's brain, but replication is neither necessary nor possible. For example, we do not assert that frames or value-range records exist in the human brain. But neither are silicon chips identical to cortical neurons. In the same way, while formal logic is seldom applied by humans to real-world problems, its spirit can be fundamentally useful in developing expert systems.

Given a body of formal assertions and implications, or propositions and production rules, there are basically two possible strategies for using them. One strategy focuses on the rules' antecedents, the other on their conclusions. They are called *forward* and *backward chaining*, respectively, and will be explained below.

A. Forward Chaining

In forward chaining, one compares the facts in the knowledge base with the antecedents of the various rules, trying to fire any rules possible. When a rule fires, its conclusions are added to the knowledge base and can potentially trigger other rules. Because attention is focused on matching antecedents against the facts that are known, this method is also called *data-driven reasoning*.

Forward chaining is often extremely useful in real-time applications. In these settings, the rules often represent event/response or condition/action knowledge. Robotics and industrial process control are examples.

B. Backward Chaining

The second strategy is somewhat more complicated. In backward chaining one starts by considering a goal, in this case a fact whose truth value is desired. The key step is to find a way of determining the truth value, which usually means finding a rule that draws a conclusion about the fact. If such a rule exists, then one can reformulate the original problem as the determination of the truth values of each of the rule's antecedents. This reformulation of prob-

lems into subproblems is iterated until each of the subproblems can be solved directly, either by finding facts in the knowledge base, or by asking questions of the user. For this reason, backward chaining is also called *goal-driven reasoning.*

C. Genie's Approach

Reasoning in the current version of Genie has a dual nature. From a holistic viewpoint, the control strategy is entirely goal-driven. Equally valid, however, is the reductionist point of view that all the system's deductions are made in data-driven mode. The two arguments are presented here.

A Genie knowledge base must contain at least one fact tagged as a *hypothesis*, or top-level goal. The inference engine records all the hypotheses and tries to verify each one in succession. Given a hypothesis, Genie looks at the **Thens~of** slot in its fact frame. This provides a list of rules that could potentially confirm the hypothesis. These rules' **Ifs** slots specify other facts which Genie treats as subgoals. The subgoals will generally have additional rules in their own **Thens~of** slots, and so on. Following all these **Thens~of** and **Ifs** links as far as they lead would generate a tree structure with facts as nodes and rules as edges. The leaves of this tree — facts that cannot be deduced from any rule in the knowledge base — constitute the information Genie must request from the user, and are consequently the simplest possible subgoals.

Consider the lowest level of deduction in this process: a rule whose antecedents are all leaves. A question will be asked for each fact, the user's answers being added to the knowledge base in a process called *memorization.* Assuming that the conditions specified by the rule's antecedents conform to the answers, these subgoals are all achieved. So the rule fires, causing its conclusions to be memorized, and a larger fraction of the problem has been solved. The procedure continues in this fashion.

In general, not all the rules will fire. A rule fails if its antecedents do not conform to the circumstances of the present run. When this happens, the desired conclusion must be achieved through other means. If the fact has untried rules in its **Thens~of**, they will be tried. If not, the fact frame will be searched for a **Default**. If all the rules that might confirm a hypothesis fail, then the hypothesis is discarded, and another one tried.

Whereas backward chaining imposes order on Genie's performance, it is forward chaining that actually deduces facts. Every time a fact is memorized, whether it was given by the user or deduced from some rule, forward chaining

is performed on it. The fact's **Ifs~of** slot lists the rules that require it. Each one of these rules that has not already fired or failed is considered. If any rule's antecedent requires the wrong truth value for the fact, then the rule fails, unless the rule frame contains a **Must~have** slot. However, if the newly memorized fact conforms with the only unfulfilled precondition of a rule, then that rule will fire. Genie will then forward chain on each of the rule's conclusions in turn.

The double linking of facts and rules makes Genie's knowledge representation scheme very flexible. Backward chaining is achieved by following the **Thens~of** and **Ifs** links. Similarly, following **Ifs~of** and **Thens** links accomplishes forward chaining. Thus, one representation of a rule can be used for both strategies. This capability is a very important one, since using only one or the other strategy has serious drawbacks. A system that only forward chains often seems to behave erratically, jumping around the knowledge base. Furthermore, a purely data-driven system cannot even ask the user any questions, since it can only passively obtain facts and apply rules to them. On the other hand, a strictly backward chaining system draws only those conclusions that are of immediate use. So it will miss drawing conclusions that are supported by its knowledge but do not lie in the direct path of its current task.

In summary, backward chaining causes Genie's reasoning to be directed from the broadest, most general conclusions, through ever more specific facts. This top-down behavior creates the impression that the system is acting purposefully. Within that context, forward chaining ensures that deductions are made as soon as the necessary knowledge is acquired.

V. USER INTERFACE

The two previous sections discussed Genie's layout and performance from an internal viewpoint. Here we shall describe the face that Genie shows to humans who interact with it at a computer terminal. First we consider the outermost layer of the program that mediates communication between the user and Genie. Then the specific modes of giving input to the system are addressed. Finally, we discuss Genie's ability to use in its communication something approaching normal English grammar. A complete specification of the input commands with which the user may control Genie's performance is provided in Appendix B.

A. Driver

Upon invoking Genie, one interacts with a function, called *Driver*, that directs Genie's operation. Anyone who invokes Driver will be categorized into one of three access classes: user, rule-writer, or programmer. The user class is the broadest. It includes those who use the system for production runs. Driver protects general users and Genie from one another, by restricting both the commands the user may use and the actions Genie may perform for the user. The rule-writer class is intended to include the knowledge engineers who maintain the system. A rule writer may execute any of the commands available to the general user. In addition, the rule writer may modify the knowledge base and dig more deeply into Genie's internal mechanisms to determine what the system is doing and why. The programmer class is the least restrictive. Programmers may execute any of the commands available to the rule writer plus several lower-level commands to debug the inference engine and peripheral components.

Driver understands a number of commands, the most important of which is *run*, the request to start up the expert system. One can also instruct Driver to show all the rules that have been applied, or all the facts or numerical values that have been derived so far in the current run. It is also possible to view rules and facts, either in an English form or in a frame form that is more like their internal representations. Concept frames can be displayed in similar fashion.

B. Run-Time Input Functions

When Genie requires information from the user, there are three schemes by which it can request it. The first and most basic of these is the simple yes/no

question. The user may respond to a yes/no question by typing one of a number of responses, each of which is equivalent to one of the responses "yes", "no", and "unknown". The three canonical responses have the effect of declaring the fact (as asked) to be true, false, or indeterminate, respectively.

The second scheme for requesting input is the menu. A menu displays a question and several numbered responses. Each of the responses represents a fact, and all the facts in a given menu form a mutually exclusive set (i.e. they are in an xor list). The user simply types in the number of the desired response. Genie then concludes that the chosen fact is true and that all others in the menu are false.

The third mode of input is the numeric question. Whenever Genie requires an arithmetic fact, rather than simply request verification of that fact, it asks for the numeric value itself. The user may respond to a numeric question in a fairly flexible algebraic notation. Possible responses include "5", "$x > 3.14$", and "$0 <= x < 100$". Once some value-range has been stored for the numeric value, all the facts that require the value are checked to determine their truths. It is assumed that the user's input was the best estimate he had of the requested value.

In addition to the permissible answers, all three question modes also allow immediate interrogation about the context in which the question is being asked. Depending on one's access class, one may type "why", to determine why the requested information is needed, "rule", to be shown the rule that is currently being tested, and "hyp", to be shown the hypothesis currently under consideration.

C. Grammar

Genie performs all its reasoning on coded representations of the domain information in the knowledge base. Generally, when it speaks to the user it must translate rules and other structures into English. To do this Genie depends on a simple yet effective pseudo-English grammar. This grammar is used in compiling the knowledge base, and then again whenever pieces of knowledge are displayed to the user.

The primary piece of knowledge that must be formulated into English is the fact. Since a proposition's assertion and its negation represent the same knowledge — in the sense that the truth value of either one follows immediately from that of the other — a single fact frame is used to represent both. Each fact frame contains a **Stmt** slot, the foundation of the formulation

process, in which is stored an affirmative English-language statement of the proposition.

So, both for building fact frames and for displaying knowledge and asking questions, the grammatical requirements are: the means to determine the sense of a statement (affirmative or negative), to switch the sense of a statement, and to turn the statement into a question. Genie accomplishes these tasks through a simple scheme that matches statements against grammatical patterns. Given the fact statement, "Compressor has driver rings", Genie will create the question, "Does compressor have driver rings?". It can complement the fact statement, "Engine is fully encased", yielding "Engine is not fully encased."

VI. DATA DIGESTION

It has been a fundamental design goal that Genie should be able to perform all of its transactions with humans in nearly natural language. When a new rule is added to a Genie knowledge base, the first form that it takes is an English-language statement. But because Genie cannot actually reason using natural language directly, there is little justification for using English strings as the medium of knowledge representation within the system. The natural-language components of Genie must be able to switch between the internal representation and statements in English.

Strong arguments *against* storing facts as strings of English words can be made from considerations of efficiency. First, it is redundant and wasteful of space to store a lengthy statement of a fact in every rule that contains the fact. But more importantly, representing the abstract object known as a fact, with all its semantic and contextual baggage, as a mere string of words is unacceptably limiting. Both goal- and data-driven reasoning require the pairing up of antecedents and conclusions of various rules. To do this with the scanty rule representation that provides only word strings requires sequentially checking every rule in the knowledge base, performing a time-consuming word-for-word check to see if facts match. Negated facts cause additional headaches.

For these reasons, Genie includes a module that preprocesses its knowledge base. Internally, rules and facts are represented by frames and referred to by unique names called *code symbols* (like "rule24" and "fact7"). The key function in the data digestion process is called *build-frames*. It reads the knowledge-base input file and puts all the static knowledge into Genie's internal representation formats. As an example, the rule frame shown in Figure 5a might have been created by build-frames from the input in Figure 8.

```
(IF    (there are struts at the front of the engine)
       (there are two flanges near the front of the engine)
       (flanges are not extremely close together)
THEN
       (engine has a front frame))
```

FIGURE 8.—*Sample Input Rule.*

The data digestion process creates rule, fact, and concept frames where necessary, recording them in tables so that, for instance, occurrences of a given fact in subsequent rules will be referred to by the same fact code. So the only

searching that Genie must do is in translating English statements into code symbols, and this is done all at once during data digestion. Other steps in data digestion are adding the links between rule and fact frames to allow chaining, and linking all the facts in each xor list. Appendix C provides a specification of the structures permissible in the knowledge-base input file.

When an end user encounters Genie, the knowledge base has already been preprocessed. The improvement in performance obtained through data digestion is marked. In a pure production system the mean time required to make one inference grows linearly as the size of the knowledge base is increased. This is so because backward chaining cannot be performed without searching the entire knowledge base for relevant rules. Digesting the rules as is done in Genie speeds the process considerably, and the time per inference is independent of the size of the knowledge base.[7]

---

7 K. Niwa, K. Sasaki, and H. Ihara. op. cit. pp. 29-36.

VII. EXPLANATION FACILITIES

A common observation among those who have designed expert systems is that a system must be accessible or it will not be used, because few will feel confident accepting the output of a black box. The program should be able to provide information about its reasoning and justify particular inferences when the user requests it to do so.[8] Genie has several facilities to provide this kind of explanation of its behavior to the user.

The simplest mechanism available to the user is the straightforward dump of current knowledge. This can take any of the following forms: listing all the facts whose **Truths** are known, listing all the rules whose **Statuses** are known, and listing all the concept attributes for which some **value-range** is known. A related mechanism is the "find" command, which finds all facts whose **Stmts** contain specified words.

The "show" command can be used to display a rule or a fact — either in English or frame form — or a concept frame. This is often useful in combination with "find".

The highest-level interrogation commands currently available to the user are "how" and "why". An example of the former is "how fact17", meaning, "By what means did you determine the truth value of fact17?". To answer the question, Genie checks the **How** slot in the fact frame, and prints an explanation of its justification for concluding the **Truth** of the fact. The "why" command asks the question, "To what end did you need that fact?" It can be used to determine Genie's motivation for asking a question. In response, Genie displays the conclusions of the rule that it is currently attempting to fire.

Taken together, these commands allow one to discover the system's lines of reasoning. This is useful for the rule writer in ensuring that rules interact to produce the intended conclusions. It is also useful for the end user in deciding whether to accept the system's conclusions and the system itself.

---

8 B. G. Buchanan and E. H. Shortliffe, eds. *Rule-Based Expert Systems.* Addison-Wesley. 1984. pp. 58-59.

VIII. FUTURE WORK

There are several potential changes that might improve both Genie's internal processes and its man/machine interface. Internally, both the reasoning mechanisms and knowledge-representation approach have been considered. There is also room for improvement in the system's interfaces with the rule writer and the user.

A. Uncertainty

There are many kinds of uncertainty inherent in any real-world problem. The omnipresence of uncertainty and imprecision is made even more bothersome by their intractability. Reasoning effectively in the face of these obstacles is among the most challenging problems in AI.

Among the types of uncertainty often encountered are simple probability of a proposition (e.g. "There is a 75 percent probability that the fragment will perforate the combustor housing."), fuzziness of a proposition (e.g. "The power turbine is very rugged."), rule strength (i.e. the extent to which the rule is applicable), and rule reliability (when knowledge is synthesized from several experts).

We have considered adding the certainty factor technique to Genie. A certainty factor is a scalar that is associated with a proposition and reflects the proposition's probability. Conclusions of rules contain certainty factors, so one rule might be said to provide stronger or weaker evidence than another rule with the same conclusion. However, parametric studies of the MYCIN program indicate that the conclusions it reaches are fairly insensitive to the choice of certainty factors. So, while certainty factors are considered helpful, careful thought will be required for their implementation.

An enticing tool is the theory of fuzzy sets, introduced by Lotfi Zadeh.[9] [10] [11]

---

9 L. A. Zadeh. "Fuzzy Sets," *Information Control*, vol. 8, 1965, pp. 338-353.

10 L. A. Zadeh. "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. smc-3, no. 1, January 1973, pp. 28-44.

11 L. A. Zadeh. *The Role of Fuzzy Logic in the Management of Uncertainty in Expert Systems.* Memorandum No. UCB/ERL M83/41, Electronics Research Laboratory, College of Engineering, University of California, Berkeley, 1983.

Fuzzy set theory promises to address many types of uncertainty. We have begun working with the Joint BRL/AMSAA Working Group on Fuzzy Sets to apply this approach to our work. Also, we are collaborating with Ronald Yager of Iona College, another pioneer in this field, on applying fuzzy logic to expert systems.[12 13]

B. Nonmonotonicity

One key difference between a human and a conventional computer program is the ability of the former to change his mind. Conceived as they are to handle imprecise knowledge, expert systems often need to modify previous beliefs. This so-called *nonmonotonic* behavior can serve two purposes.

In some applications, knowledge is time variant. Under these circumstances, a newly acquired bit of information might supersede and contradict an earlier decision. This can require undoing a substantial fraction of the system's reasoning, and it is difficult to ensure that no conflicts appear. We have not tried to equip Genie to handle this class of problem, since in the applications considered so far one's information does not become any more complete or correct during a run, so what the user tells Genie can be treated as constant.

Another use for nonmonotonicity is in making tentative conclusions. For example, if a system does not know the truth value of a proposition, it might assume a value and proceed. Any conclusions eventually reached must, of course, be flagged as depending on the assumption. Or a system can employ proof by contradiction. If one choice of truth value for a given proposition leads to a contradiction, then that truth value can be excluded, and the opposite truth value inferred. Both of these methods could be fairly easily added to Genie, using a mechanism such as OPS5's time stamps.

---

12 R. R. Yager. "Querying Knowledge Base Systems with Linguistic Information via Knowledge Trees," *International Journal of Man-Machine Studies*, 19, 1983, pp. 73-95.

13 R. R. Yager. "An Approach to Inference in Approximate Reasoning," *International Journal of Man-Machine Studies*, 13, 1980, pp. 323-338.

C. Higher-Level Organization of Knowledge

The exclusive use of rules to represent domain knowledge makes it impossible to capture any but the simplest patterns in the knowledge. More abstract organization can only be represented by more complex structures. In order for Genie to reason more powerfully, explain its behavior at a suitably high level, and be appropriate as a teaching tool, it must have a fair level of abstraction. This consideration is central to the enhancement of Genie's performance.

The first two steps toward higher-level organization were the representation of rules and facts as frames, and the use of concept frames. Concept frames are currently used only for numeric values in arithmetic facts. The next step is probably to use a more robust concept frame, like that found in Lenat's program, AM.[14] [15] Thus, in a future incarnation Genie might build concept frames for every "object" mentioned in the rule base, so it could fulfill requests like "Show every rule that mentions annular combustors".

Of course, implementing this capability would require that the input rules be analyzed using a much more powerful grammar than the one with which Genie is currently endowed. Genie cannot now be said to understand its knowledge base in abstract terms: it breaks rules into facts and very effectively handles the relations among them, but on fact statements it only performs surface-level manipulations. Clearly, a deeper understanding would require more intelligence in interpreting the knowledge.

D. Rule Writer

As a rule base grows, keeping it free of conflicts, contradictions, and overlaps becomes extremely difficult. While building individual rules is clear and straightforward, ensuring that the integration of hundreds of rules produces the desired results can be a problem. It would be a great advantage to have a component that helped the knowledge engineer manage the development and maintenance of the knowledge base, facilitating addition and debugging of rules.

---

14 D. Lenat. *AM: An artificial intelligence approach to discovery in mathematics as heuristic search.* SAIL AIM-286, Stanford Artificial Intelligence Laboratory, 1976.

15 D. A. Waterman and F. Hayes-Roth, eds. *Pattern-Directed Inference Systems.* Academic Press, Inc., 1978. pp. 30-33.

Like most of the enhancements discussed here, a rule writing tool's utility depends on its intelligence. For example, even the ability to recognize when two similar facts have related meanings is difficult to automate. A first-pass rule writer could be made to compile rules into the knowledge base as they were entered. Such a program would be able to determine, just as Genie does now, which rules use given facts.

E. More Natural User Interface

Genie ought to be able to converse with the user in something closer to normal English. This is tied in with increasing the order in the knowledge base, since sophisticated statements are built from and reflect elaborate knowledge structures.

Another major improvement in the user interface will be possible when Genie is moved to a LISP machine in the near future. These single-user work stations provide a remarkably powerful environment for both development and production runs. A combination of mouse, windows, pop-up menus, and high-resolution graphics will make communication simple and quite fast.

APPENDIX A

Sample Run of Tess

(Turbine Engine Expert System)

VLD Turbine Expert System

Please wait while Genie version 4.0 is loaded.
Please wait while Genie loads the data.

123 rule frames loaded.
287 fact frames loaded.
4 menu frames loaded.
7 concept frames loaded.
6 exclusive-or sets loaded.
9 potential hypotheses loaded.
4 'ask-first' facts loaded.
Fact symbol table loaded.

All data has been loaded.

Hello, Don.
Give me a command-- **help**

Possible commands are:
how <factcode> why <factcode>
find <one or more words in a fact>
(any lisp command) showvar <variable>
show <rulecode> show <factcode>
showframe <rulecode> or <factcode> or <menucode>
showconcept [<concept> [<attribute>]]
rules facts numbers
run bye help

Give me a command-- **run**

Complete the following statement

Engine type is:

1 turbo shaft
2 turbo jet
3 turbo fan

Enter the number of your choice-- **1**

Complete the following statement

Detail level is:

1 actual engine
2 pictures and drawings
3 intelligence information

Enter the number of your choice-- **1**
What is the horsepower of shaft? **1400**
Does compressor have driver rings? **why**

BECAUSE
this question is used in determining if:
compressor has variable geometry

Does compressor have driver rings? **yes**
Rule121 deduces compressor has variable geometry.

What is the number of stages? **show rule121**

IF [fact280] compressor has driver rings [succeeded]
and [fact281] there are vane lever arms [unknown]
NEED at least 1 of these facts
THEN [fact54] compressor has variable geometry [succeeded]

What is the number of stages? **find fixed geometry**

[fact7] : compressor has fixed geometry

What is the number of stages? **how fact7**

Genie learned that: compressor does not have fixed geometry
because it was mutually exclusive with:
[fact54] compressor has variable geometry

What is the number of stages? **6**

What is the diameter of inlet? **facts**

given	: [fact5]	: engine type is turbo shaft
excluded	: [fact287]	: engine type is not turbo fan
excluded	: [fact286]	: engine type is not turbo jet
given	: [fact6]	: detail level is actual engine
excluded	: [fact47]	: detail level is not intelligence information
excluded	: [fact46]	: detail level is not pictures and drawings
deduced	: [fact55]	: T53/L13 shaft hp
deduced	: [fact134]	: T58/5 shaft hp
deduced	: [fact250]	: T700/700 shaft hp
given	: [fact280]	: compressor has driver rings
deduced	: [fact54]	: compressor has variable geometry
excluded	: [fact7]	: compressor does not have fixed geometry
indeterminate	: [fact161]	: intelligence info indicates T64/3
indeterminate	: [fact162]	: pictures and drawings indicate T64/3
indeterminate	: [fact163]	: engine inspection indicates T64/3
elim (hyp)	: [fact164]	: compressor does not resemble T64/3

What is the diameter of inlet? **16 <= diameter < 18**

. . . *user supplied more information*

What is the compression-ratio of compressor? **6.7**
Rule18 deduces engine inspection indicates T53/L13.
Rule17 deduces compressor resembles T53/L13.

Final conclusion: [fact53] : Compressor resembles T53/L13.

Give me a command-- **numbers**

given : 16 <= diameter of inlet < 18
given : speed of compressor = 25400
given : compression-ratio of compressor = 6.700000
given : overall-length of engine = 47.599998
given : dry-weight of engine = 535
given : airflow of system = 12.200000
given : horsepower of shaft = 1400
given : number of stages = 6

Give me a command-- **bye**
Goodbye, Don.

APPENDIX B

List of Run-Time Key Words

Commands which may be given only to the top level driver

run:
purpose: run the expert system
form: **run**

bye:
purpose: terminate Genie and exit
form: **bye**

Responses which may be given only in reply to a question

simple answer:
purpose: answer to a question that Genie asked
restrictions: depends on the type of question
examples: **yes**
no
unknown
18.7
5 < height <= 6.3

why:
purpose: ask Genie to explain why it is posing this question
form: **why**

hyp:
purpose: ask Genie what the current hypothesis is
form: **hyp**

rule:
purpose: ask Genie what the current rule is
form: **rule**

Commands which may be given at any time

help:
- purpose: display table of allowable commands and responses
- form: **help**

how:
- purpose: ask Genie how a fact was learned
- form: **how** <factcode>
- example: **how** fact52

find:
- purpose: get fact codes that contain certain words
- form: **find** <list of one or more words>
- example: **find** compressor housing

show:
- purpose: ask Genie to display a rule or fact in English
- forms: **show** <rulecode>
  show <factcode>
- examples: **show** rule83
  show fact125

rules:
- purpose: ask Genie to display all rules that have been used
- form: **rules**

facts:
- purpose: ask Genie to display all facts that have been learned
- form: **facts**

numbers:
- purpose: ask Genie to display all numbers that have been learned
- form: **numbers**

showframe:
- purpose: ask Genie to display a rule, fact, or menu as a frame
- forms: **showframe** <rulecode>
  showframe <factcode>
  showframe <menucode>
- examples: **showframe** rule83
  showframe fact125
  showframe menu2

showconcept:

purpose: ask Genie to display a concept frame
form: **showconcept** [<concept> [<attribute>]]
example: **showconcept** shaft horsepower

showvar:

purpose: ask Genie to evaluate a LISP variable
form: **showvar** <variable>
restrictions: only executable by special users
example: **showvar** HypCodeList

any LISP command:

purpose: execute a LISP command
form: (function arguments)
restrictions: only executable by special users
example: (fpp 'menu2)

APPENDIX C

List of Knowledge-Base Key Words

IF:

usage: define an if/then rule

form: (**IF** FactStmt FactStmt ...
[**NEED** (Number)]
THEN FactStmt FactStmt ...)

restrictions: rules may not be logically recursive

examples: (**IF** (detail level is intelligence information)
(number of stages GT 8)
THEN (intelligence info indicates axial compressor))
(**IF** (compressor has driver rings)
(there are vane lever arms)
NEED (1)
THEN (compressor has variable geometry))

XOR:

usage: declare facts to be mutually exclusive
if one fact is true the rest will be memorized as false

form: (**XOR** PosFact PosFact ...)

restrictions: facts must be positive

example: (**XOR** (compressor has variable geometry)
(compressor has fixed geometry))

HYP:

usage: declare a fact to be a hypothesis

form: (**HYP** PosFact PosFact ...)

restrictions: facts must be positive

example: (**HYP** (compressor resembles a T58)
(compressor resembles a J57))

QUESTION:

usage: define question to be used instead of generating one

form: (**QUESTION** PosFact Question)
(**QUESTION** (A of C) Question)

restrictions: fact must be positive

examples: (**QUESTION** (there is diameter decrease before compressor)
(Is there a significant decrease in the diameter
before the compressor?))
(**QUESTION** (number of stages)
(How many stages are in the compressor?))

MENU:

usage:	define multiple choice fact menu
form:	(**MENU** Template [Question] (Choice Choice ...))
restrictions:	facts must be positive
examples:	(**MENU** (compressor housing is X) ((mild steel) (aluminum)))
	(**MENU** (detail level is X) (What is the level of detail?) ((low) (medium) (high)))

ASK-FIRST:

usage:	force Genie to verify (ask) these facts first
form:	(**ASK-FIRST** FactStmt FactStmt ...)
restrictions:	none
example:	(**ASK-FIRST** (compressor has variable geometry) (number of stages))

NOSHOW:

usage:	do not tell user when this fact is deduced
form:	(**NOSHOW** PosFact PosFact ...)
restrictions:	facts must be positive
example:	(**NOSHOW** (T58 inlet-diameter) (T58 compression-ratio))

DEFAULT:

usage:	declare truth to be assumed if fact cannot be proven
form:	(**DEFAULT** (PosFact FactStmt) (PosFact FactStmt) ...)
restrictions:	none, but FactStmt will usually be negative
examples:	(**DEFAULT** ((compressor has variable geometry) (compressor does not have variable geometry)))

NULL-HYP:

usage:	define statement to be displayed if no solution found
form:	(**NULL-HYP** Statement)
restrictions:	none
example:	(**NULL-HYP** (Data does not match any known compressor.))

DISTRIBUTION LIST

No. of Copies	Organization
12	Administrator Defense Tech Info Ctr ATTN: DTIC-DDA Cameron Station Alexandria, VA 22304-6145
1	Director Inst for Def Analyses 1818 Beauregard St. Alexandria, VA 22311
1	Director Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, VA 22209
1	HQDA (DAMA-ART-M) Washington, DC 20310
1	HQDA (DAMA-ARR) Washington, DC 20310-0632
1	HQDA (DACA-CW) Washington, DC 20310
1	HQDA (DAMI) Washington, DC 20310
1	Commander USA LABCOM ATTN: AMSLC-TD 3800 Powder Mill Rd. Adelphi, MD 20783-1145
1	Director US Army Engineer Waterways Experiment Station ATTN: Mr. D. Colthorp P. O. Box 631 Vicksburg, MS 39108
1	Commander US Army Materiel Command ATTN: AMCDRA-ST 5001 Eisenhower Avenue Alexandria, VA 22333
1	Commander US Army Materiel Command ATTN: AMCLD 5001 Eisenhower Avenue Alexandria, VA 22333-0001
1	Commander Armament R&D Center US Army AMCCOM ATTN: SMCAR-TDC Dover, NJ 07801
1	Commander Armament R&D Center US Army AMCCOM ATTN: SMCAR-TSS Dover, NJ 07801
1	Commander Armamment R&D Center US Army AMCCOM ATTN: SMCAR-LC Dover, NJ 07801
1	Commander US Army Armament, Munitions & Chem Com ATTN: SMCAR-ESP-L Rock Island, IL 61299
1	Commander US Army Armament, Munitions & Chem Com ATTN: AMSAR-SA, Mr. Michels Rock Island, IL 61299

DISTRIBUTION LIST

No. of Copies	Organization
1	Director Benet Weapons Laboratory Armament R&D Center US Army AMCCOM ATTN: SMCAR-LCB-TL Watervliet, NY 12189
1	Commander US Army Aviation Research and Development Command ATTN: AMSAV-E 4300 Goodfellow Blvd St. Louis, MO 63120
1	Commander US Army Air Mobility R&D Laboratory Ames Research Center Moffett Field, CA 94035
1	Director Appl. Tech. Directorate USAARTA (AVSCOM) ATTN: DAVDL-EU-SY-RPV Fort Eustis, VA 23604
1	Commander US Army Troop Support and Aviation Materiel Readiness Command ATTN: AMSTS-G 4300 Goodfellow Boulevard St. Louis, MO 63120
1	Commander US Army Communications-Electronics Command ATTN: AMSEL-ED Fort Monmouth, NJ 07703
1	Commander US Army Communications Command ATTN: ATSI-CD-MD Fort Huachuca, AZ 85613
1	Commander ERADCOM Tech. Library ATTN: DELSD-L(Rpts. Sec) Fort Monmouth, NJ 07703-5301
1	Commander US Army Missile Command Res, Dev & Eng Center ATTN: AMSMI-RD Redstone Arsenal, AL 35898
1	Director US Army Missile and Space Intel Center ATTN: AIAMS-YDL Redstone Arsenal, AL 35898-5500
1	Commander US Army Belvoir R&D Center ATTN: AMDME-WC Fort Belvoir, VA 22060-5606
1	Commander US Army Tank Automotive Command ATTN: AMSTA-TSL Warren, MI 48090

DISTRIBUTION LIST

No. of Copies	Organization
2	Commander US Army Research Office ATTN: AMXRO-MA, Dr. J. Chandra Dr. R. Launer P. O. Box 12211 Research Triangle Park, NC 27709-2211
1	Commander US Army Combat Dev & Experimentation Command ATTN: ATEC-SA, Dr. M. R. Bryson Fort Ord, CA 93941
1	Commander US Army Harry Diamond Laboratory ATTN: DELHD-RT-RD 2800 Powder Mill Rd Adelphi, MD 20783
1	Commander US Army Logistics Center ATTN: ATLC-OOM, J. Knaub Ft. Lee, VA 23801
1	Director US Army Concepts Analysis Agency ATTN: CSCA-AST, Mr. C. Bates 8120 Woodmont Avenue Bethesda, MD 20014
1	Director USA Res, Dev and Standardization Group (UK ATTN: Dr. J. Gault Box 65 APO New York, NY 09510
1	Director US Army Concepts Analysis Agency ATTN: CSCA-AST, Mr. B. Graham 8120 Woodmont Avenue Bethesda, MD 20014
1	Director Walter Reed Army Institute of Research Walter Reed Army Medical Center ATTN: SGRD-UWE, Dr. D. Tang Washington, DC 20012
1	President US Army Airborne, Electronics & Special Warfare Board Fort Bragg, NC 28307
1	President US Army Armor & Engineer Board Fort Knox, KY 40121
1	President US Army Artillery Board Fort Sill, OK 73503
1	AFWL/SUL Kirtland AFB, NM 87117-6008
1	AFWL/NTES(Dr. Ross) Kirtland AFB, NM 87117-6008
1	Commander US Army Materials and Mechanics Research Ctr Watertown, MA 02172

DISTRIBUTION LIST

No. of Copies	Organization
1	Commander US Army Training and Doctrine Command Fort Monroe, VA 23651
1	Commander US Army TRADOC Systems Analysis Activity ATTN: ATAA-SL, Tech Lib White Sands Missile Range NM 88002
2	Commandant US Army Armor School ATTN: Armor Agency ATSB-CD-MM Fort Knox, KY 40121
1	Commandant US Army Artillery School ATTN: ATSF-CA, Mr. Minton Fort Sill, OK, 73503
1	Commandant US Army Aviation School ATTN: Aviation Agency Fort Rucker, AL 36360
1	Commandant US Army Infantry School ATTN: ATSH-CD-CSO-OR Fort Benning, GA 31905
1	Commandant US Army Infantry School ATTN: ATSH-CD-CSO-OR, Mr. J. D'Errico Fort Benning, GA 31905
1	Commandant US Army Intelligence Sch ATTN: Intel Agcy Fort Huachuca, AZ 85613
1	Commander US Army Development & Employment Agency ATTN: MODE-TED-SAB Fort Lewis, WA 98433
1	Chief of Naval Operations ATTN: OP-721 Department of the Navy Washington, DC 20350
2	Commander Naval Air Systems Command ATTN: WEPS, Mr. R. Sawyer AIR-604 Washington, DC 20360
1	Commander Naval Air Development Center, Johnsville ATTN: Code SRS Warminster, PA 18974
1	Commander Naval Surface Weapons Ctr ATTN: DX-21, Lib Br. Dahlgren, VA 22448
1	Commander Naval Surface Weapons Cen ATTN: Code G31 Dahlgren, VA 22448
1	Commander Naval Weapons Center ATTN: Code 31804 China Lake, CA 93555
1	Commander Naval Research Lab Washington, DC 20375

DISTRIBUTION LIST

No. of Copies	Organization
1	Commander David Taylor Naval Ships Research & Development Center ATTN: Tech Library Bethesda, MD 20084
1	Commandant US Marine Corps ATTN: AAW-1B Washington, DC 20380
1	Commandant US Marine Corps ATTN: POM Washington, DC 20380
1	Commanding General Fleet Marine Force, Atlantic ATTN: G-4 (NSAP) Norfolk, Va 23511
1	Commander Marine Corps Development and Education Command (MCDEC) Quantico, VA 22134
1	HQ USAF/SAMI Washington, DC 20330-5425
3	AFSC (SCFO; SDW; DLCAW) Andrews AFB, MD 20331
2	ADTC (DLODL; ADBRL-2) Eglin AFB, FL 32542
1	AFATL (DLMM) Eglin AFB, FL 32542
1	USAFTAWC/ADTC Eglin AFB, FL 32542
1	Air Force Armament Lab ATTN: AFATL/DLODL Eglin AFB, FL 32542-5000
1	TAC (INAT) Langley AFB, VA 23665
1	AFWAL/FIBC Wright-Patterson AFB, OH 45433
1	FTD (ETD) Wright-Patterson AFB, OH 45433
1	Department of State Office of Security 21-st and C Street Wash. DC 20013
10	Central Intel Agency Office of Central Ref Dissemination Branch Room GE-47 HQS Washington, DC 20502
1	Battelle Columbus Laboratories ATTN: Ordnance Div 505 King Avenue Columbus, OH 43201
2	Southwest Research Inst Dept of Mech Sciences ATTN: Mr. A. Wenzel Mr. P. Zabel 8500 Culebra Road San Antonio, TX 78284

DISTRIBUTION LIST

No. of Copies	Organization
1	Oklahoma State University Field Office ATTN: Mrs. Ann Peebles P. O. Box 1925 Eglin Air Force Base, FL 32542
1	Prof. Lotfi Zadeh Dept of Electrical Eng & Comp Science University of California Berkeley, CA 94720
1	Prof. James T.P. Yao School of Civil Eng Civil Eng Bldg Purdue University West Lafayette, In 47907
1	Dr. William H. Friedman Dept. Econ & Dec. Sci. Loyola College 4501 N. Charles St Baltimore, MD 21210-2699
1	Prof. J.L.Chameau Geotech Eng Grissom Hall Purdue University West Lafayette, IN 47907
1	Machine Intelligence Inst Iona College ATTN: Dr.R. Yager New Rochelle, NY 10801
1	Dr. Felix S. Wong Weidlinger Associates 620 Hansen Way Suite 100 Palo Alto, CA 94304
1	Dr. Steven B. Boswell MIT Lincoln Lab, Group 94 Lexington, MA 02173-0073
	Aberdeen Proving Ground
12	Dir, USAMSAA ATTN: AMXSY-D, Mr. K. Myers AMXSY-MP, Mr. H. Cohen AMXSY-A, Mr. D. O'Neill AMXSY-RA, Mr. R. Scungio AMXSY-GS, Mrs. M. Ritondo Dr. M. Starks AMXSY-AAG, Mr. W. Nicholson Mr. C. Abel AMXSY-G Mr. J. Kramar AMXSY-J, Mr. J. Blomquist Mr. J. Matts AMXSY-LR, Mr. W. Webster
1	Cdr, USATECOM ATTN: AMSTE-TO-F
3	Cdr, CRDC, AMCCOM ATTN: SMCCR-RSP-A SMCCR-MU SMCCR-SPS-IL

USER EVALUATION SHEET/CHANGE OF ADDRESS

This Laboratory undertakes a continuing effort to improve the quality of the reports it publishes. Your comments/answers to the items/questions below will aid us in our efforts.

1. BRL Report Number_____ Date of Report_____

2. Date Report Received_____

3. Does this report satisfy a need? (Comment on purpose, related project, or other area of interest for which the report will be used.)_____

4. How specifically, is the report being used? (Information source, design data, procedure, source of ideas, etc.)_____

5. Has the information in this report led to any quantitative savings as far as man-hours or dollars saved, operating costs avoided or efficiencies achieved, etc? If so, please elaborate._____

6. General Comments. What do you think should be changed to improve future reports? (Indicate changes to organization, technical content, format, etc.)

CURRENT ADDRESS

Name

Organization

Address

City, State, Zip

7. If indicating a Change of Address or Address Correction, please provide the New or Correct Address in Block 6 above and the Old or Incorrect address below.

OLD ADDRESS

Name

Organization

Address

City, State, Zip

(Remove this sheet along the perforation, fold as indicated, staple or tape closed, and mail.)

FOLD HERE

Director
U.S. Army Ballistic Research Laboratory
ATTN: SLCBR-DD-T
Aberdeen Proving Ground, MD 21005-5066

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE, $300

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO 12062 WASHINGTON, DC
POSTAGE WILL BE PAID BY DEPARTMENT OF THE ARMY

Director
U.S. Army Ballistic Research Laboratory
ATTN: SLCBR-DD-T
Aberdeen Proving Ground, MD 21005-9989

FOLD HERE

END

DTIC

9-86